python for Computational Problem Solving - pCPS - Control Structures Lecture Slides - Class #13 to Class#14

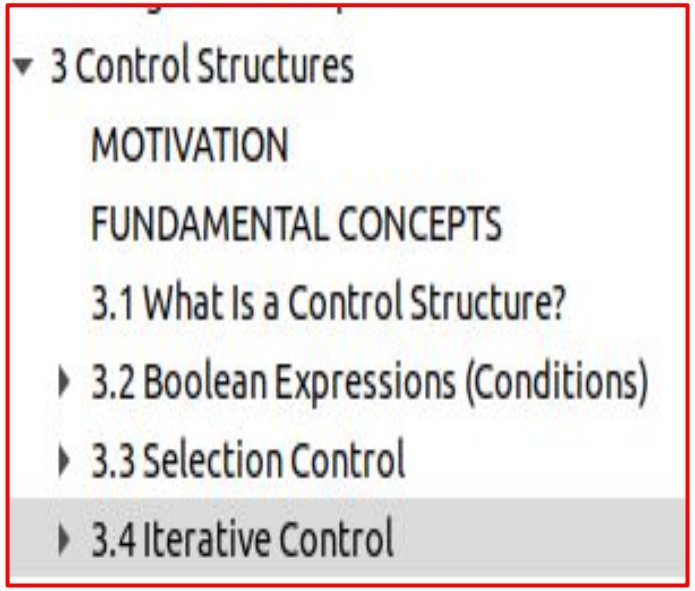
Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

python for Computational Problem Solving Syllabus

Unit I: Computational Problem Solving - 12 Hours

Limits of Computational Problem Solving - Computer Algorithm - Computer Hardware - Digital Computer - Operating System- Limits of IC technology - Computer Software - Syntax, semantics and program translation ,Introduction to Python Programming Language, IDLE Python Development Environment, Output function - variables, types and id,input function , operators and expressions, Control structures .

T1: 1.1 – 1.7
T1: 2.1 - 2.4
T1: 3.1 – 3.4



pCPS 3.4 Iterative Control

- An **iterative** control statement is a control statement providing the **repeated** execution of a **set** of **instructions**.

- An iterative control structure is a set of instructions and the iterative control statement(s) **controlling** their execution.

- Because of their repeated execution, iterative control structures are commonly referred to as "**loops**."

pCPS 3.4.1 Iterative Control - while statement

```
#  Iterative_Control examples

n = int(input('How many First n Natural Numbers you want me to sum ? '))

Counter = 1
Sum = 0

while(Counter<=n):
    print('Counter',Counter,sep='=',end='\t')
    Sum=Sum+Counter
    print('Sum',Sum,sep='=')
    Counter+=1


print('The Summation of First {0:5d} natural numbers is {1:10d}'.format(n,Sum))
```

```
How many First n Natural Numbers you want me to sum ? 10
Counter=1       Sum=1
Counter=2       Sum=3
Counter=3       Sum=6
Counter=4       Sum=10
Counter=5       Sum=15
Counter=6       Sum=21
Counter=7       Sum=28
Counter=8       Sum=36
Counter=9       Sum=45
Counter=10      Sum=55
The Summation of First    10 natural numbers is         55
```

pCPS 3.4.1 Iterative Control - while statement

```
#  Iterative_Control examples

n = int(input('How many First n Odd Numbers you want me to print ? '))

Counter = 0
Odd = 0


print('\nFirst {0:3d} odd Numbers'.format(n))
while(Counter<=n-1):
    print(Odd,end=' ')
    Odd=2*Counter+1
    Counter+=1
```

```
How many First n Odd Numbers you want me to print ? 100

First 100 odd Numbers
0 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79
81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139 141 14
3 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197
```

pCPS 3.4.1 Iterative Control - while statement

```
#  Iterative_Control examples
# range Demonstration

Ranges = int(input('Different Ranges needed'))

Counter = 0
Steps = 1
while (Counter <=Ranges):
      print(range(0,Counter,Steps))
      Counter+=1
```

```
Different Ranges needed10
range(0, 0)
range(0, 1)
range(0, 2)
range(0, 3)
range(0, 4)
range(0, 5)
range(0, 6)
range(0, 7)
range(0, 8)
range(0, 9)
range(0, 10)
```

pCPS 3.4.1 Iterative Control - while statement

```
#  Iterative_Control examples
# range Demonstration

Ranges = int(input('Different Ranges needed'))

Counter = 0
Steps = 2
while (Counter <=Ranges):
    print(range(0,Counter,Steps))
    Counter+=1
```

```
Different Ranges needed10
range(0, 0, 2)
range(0, 1, 2)
range(0, 2, 2)
range(0, 3, 2)
range(0, 4, 2)
range(0, 5, 2)
range(0, 6, 2)
range(0, 7, 2)
range(0, 8, 2)
range(0, 9, 2)
range(0, 10, 2)
```

pCPS 3.4.1 Iterative Control - while statement

```
: n = int(input('Input ? '))
  Counter = 0
  while(Counter<=range(0,n)):
      print(range(Counter,n))
      Counter+=1
```

```
Input ? 10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_15742/1233177986.py in <module>
      1 n = int(input('Input ? '))
      2 Counter = 0
----> 3 while(Counter<=range(0,n)):
      4     print(range(Counter,n))
      5     Counter+=1

TypeError: '<=' not supported between instances of 'int' and 'range'
```

pCPS 3.4.1 Iterative Control - while statement

```
n = int(input('Input ? '))
Counter = 0
while Counter in range(0,n):
    print(Counter,end=' ')
    Counter = Counter + 1
```

```
Input ? 100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
```

pCPS 3.4.2 Input Error Checking

- The while statement is well suited for input error checking in a program.

```
Option = input('To Proceed Yes / No::')
Attempt =  1
Flag = True

while(Option!='Yes'and Attempt<5):
    print('Attempt#',Attempt)
    Option = input('To Proceed Yes / No::')
    Attempt=Attempt+1


if (Attempt<=5) and (Option=='Yes'):
    print('Sucessful Attempt at #',Attempt)
else:
    print('Possible Malicious Attempt')
```

```
To Proceed Yes / No::Try
Attempt# 1
To Proceed Yes / No::Try
Attempt# 2
To Proceed Yes / No::Yes
Sucessful Attempt at # 3
```

```
Option = input('To Proceed Yes / No::')
Attempt =  1
Flag = True

while(Option!='Yes'and Attempt<5):
    print('Attempt#',Attempt)
    Option = input('To Proceed Yes / No::')
    Attempt=Attempt+1


if (Attempt<=5) and (Option=='Yes'):
    print('Sucessful Attempt at #',Attempt)
else:
    print('Possible Malicious Attempt')
```

```
To Proceed Yes / No::Try
Attempt# 1
To Proceed Yes / No::Try
Attempt# 2
To Proceed Yes / No::Try
Attempt# 3
To Proceed Yes / No::Try
Attempt# 4
To Proceed Yes / No::Try
Possible Malicious Attempt
```

pCPS 3.4.3 Infinite Loops

- An **infinite** loop is an iterative control structure that **never terminates** or eventually **terminates** with a **system error**.

- **Infinite loops** are generally the **result** of **programming errors**.

```
In [*]: Option = input('To Proceed Yes / No::')
        Attempt =  1
        Flag = True

        while(Option!='Yes'and Attempt<5):
            print('Attempt#',Attempt)
            Option = input('To Proceed Yes / No::')

        if (Attempt<=5) and (Option=='Yes'):
            print('Sucessful Attempt at #',Attempt)
        else:
            print('Possible Malicious Attempt')
```

```
To Proceed Yes / No::1
Attempt# 1
To Proceed Yes / No::2
Attempt# 1
To Proceed Yes / No::4
Attempt# 1
To Proceed Yes / No::4
Attempt# 1
To Proceed Yes / No::5
Attempt# 1
To Proceed Yes / No::5
Attempt# 1
To Proceed Yes / No::5
Attempt# 1
To Proceed Yes / No::5
Attempt# 1
To Proceed Yes / No::5
Attempt# 1
To Proceed Yes / No::
Attempt# 1
```

- *** indicates infinite loop since Attempt is not modified with in the loop for a non Yes option**

pCPS 3.4.4 Definite vs. Indefinite Loops

- A **definite** loop is a program loop in which the **number** of **times** the loop will **iterate** can be determined **before** the **loop** is **executed**.

```
# Demonstration of definite loops

n = int(input('How many times you want me to loop (Enter >=0) ? '))

Loop = 0
Sum = 0

while(Loop<n):
    Loop+=1


print('Looped ', Loop,' Times')
```

```
How many times you want me to loop (Enter >=0) ? 0
Looped  0  Times
```

```
# Demonstration of definite loops

n = int(input('How many times you want me to loop (Enter >=0) ? '))

Loop = 0
Sum = 0

while(Loop<n):
    Loop+=1


print('Looped ', Loop,' Times')
```

```
How many times you want me to loop (Enter >=0) ? 10
Looped  10  Times
```

pCPS 3.4.4 Definite vs. Indefinite Loops

- An **indefinite** loop is a program loop in which the number of times that the loop will iterate **cannot** be **determined** before the **loop** is executed.

```
# Demonstration of indefinite loops
Loop = 0
Option = input('To Proceed Yes / No::')
Attempt =  1
Flag = True

while(Option!='No'):
    Loop=Loop+1
    Option = input('To Proceed Yes / No::')


print('Looped {0} number of times'.format(Loop))
```

```
To Proceed Yes / No::Yes
To Proceed Yes / No::Yes
To Proceed Yes / No::Yes
To Proceed Yes / No::No
Looped 3 number of times
```

```
# Demonstration of indefinite loops
Loop = 0
Option = input('To Proceed Yes / No::')
Attempt =  1
Flag = True

while(Option!='No'):
    Loop=Loop+1
    Option = input('To Proceed Yes / No::')


print('Looped {0} number of times'.format(Loop))
```

```
To Proceed Yes / No::Yes
To Proceed Yes / No::Yes
To Proceed Yes / No::Yes
To Proceed Yes / No::Yes
To Proceed Yes / No::Yes
To Proceed Yes / No::No
Looped 5 number of times
```

pCPS 3.4.5 Boolean Flags and Indefinite Loops

- Often the condition of a given while loop is denoted by a **single** Boolean variable, called a Boolean **flag** .

```
Loop=0
Flag = True
while(Flag):
    Loop+=1
    print('Loop#',Loop)
    if (Loop>=10):
        Flag=False
```

```
Loop# 1
Loop# 2
Loop# 3
Loop# 4
Loop# 5
Loop# 6
Loop# 7
Loop# 8
Loop# 9
Loop# 10
```

```
Loop=0
Flag = False
while(Flag):
    Loop+=1
    print('Loop#',Loop)
    if (Loop>=10):
        Flag=False
```

End of Unit 1

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University
nitin.pujari@pes.edu

For Course Digital Deliverables visit www.pesuacademy.com